Every Child a Coder? Research Challenges for a 5-18 Programming Curriculum

Kate Howland
Department of Informatics
University of Sussex
k.l.howland@sussex.ac.uk

Judith Good
Department of Informatics
University of Sussex
j.good@sussex.ac.uk

Judy Robertson
Moray House School of Education
University of Edinburgh
judy.robertson@ed.ac.uk

Andrew Manches
Moray House School of Education
University of Edinburgh
a.manches@ed.ac.uk

Abstract

The current drive in many countries to teach computing, particularly programming, to all from an early age, has potential to empower and support children in creative and problem-solving tasks. However, there are a number of challenges in ensuring that computing curricula, tools and environments embody appropriate progression and engender motivation for the topic across the school years. This workshop will consider the key research challenges in learning coding throughout childhood, with contributions from developmental psychologists, educators, researchers of children's programming, and designers of developmentally appropriate technologies for children.

Author Keywords

Children; coding; programming; design

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

Introduction

In 2014, the English National Curriculum introduced Computing as a new subject, requiring that children be taught coding, algorithms and other computational concepts from age 5 [6], and there are calls for similar moves across Europe [10] and the USA [4]. As well as addressing the shortages in high-level computing students and an appropriately trained workforce, the new curriculum aims to give children knowledge, understanding and skills that will transfer to other subject areas and everyday life. Yet there are a number of challenges in these laudable aims, notably how programming and computational thinking can be taught in a way which engenders motivation, is developmentally appropriate, and has a clear, joined-up, progression across ages. Although there is extensive

research on how to teach programming, and broader computer science concepts, to novices, this has primarily focused on older children, and on learners in further or higher education (see, e.g.,[16]). At present we have limited knowledge about young people's abilities to engage with computational concepts at different ages. Furthermore, we know very little about the learning pathway from age 5, and,how children's understanding of computational concepts develops and deepens over time, with the same holding true for children's motivation for programming.

This workshop seeks to draw together existing theoretical and empirical work across ages in this area, in order to identify the key questions which should define a future research agenda.

Background

From a theoretical perspective, given the importance of abstraction and logical thinking for computation, Piagetian theory may potentially be of relevance. In his four stages of development, Piaget maintained that children have the capacity for abstract and logical thinking only in the final "formal operational" stage (age 11/12+). At earlier stages, their capacity for logic and abstract thought is thought to be non-existent ("pre-operational" stage, age 3-7), or only operates on physical objects ("concrete operational", age 7-11) [12]. However, neo-Piagetian theory suggests that progression through the four stages is not generalised or explicitly tied to age, and is instead a function of the level of expertise in a specific problem domain. Capacity for abstraction can then be linked to the time devoted to developing expertise in a given area [3].

The significance of theoretical arguments surrounding the role of concrete experience in children's capacity to reason in this domain is illustrated by designs aiming to introduce computing concepts through more tangible forms of interaction. For example, Wyeth and colleagues designed *Electronic Blocks,* a tangible programming tool, to take into account pre-operational children's reliance on sensorimotor interaction, and their propensity to learn through exploratory play and construction tasks [17]. The TangibleK Robotics program, developed by Bers et al. [1], also builds on children's interaction with the physical world using tangible interfaces such as CHERP, to support computational learning for children as young as 3 . Furthermore, Bers and colleagues have started to map out computational thinking learning trajectories and develop appropriate curriculum materials [1].

In parallel with developmental research in the area, a number of other tools aimed at young children have been produced, for example, Cricket [15], Curly Bot [8], Topobo [13] and ScratchJr [7]. Tools aimed at older children (aged 11+) focus less on tangible and physical interaction, and more on the use of concrete representations in the form of virtual worlds, in addition to the motivational aspects of context and task. For example, the Flip programming language is embedded in a game creation toolset so that programming is introduced as a way of customising the behaviour of characters and objects in a game world, which young people have themselves created [9]. Similarly, tools such as Scratch [14], Alice [5], and Greenfoot [11] provide split-screen interfaces which show the code alongside a game world or animation which it controls.

However, despite the proliferation of tools, and existing research in the field, there remains a need to draw together what we know about children's programming and link it in a cohesive manner. Doing so will provide a clearer understanding of the developmental, cognitive, and motivational aspects of learning programming throughout childhood. This will in turn allow us to identify gaps in our knowledge and conduct further research (potentially using existing tools and environments). These findings can then feed into the design of a comprehensive curriculum which incrementally builds on children's knowledge, and may even start before the advent of formal schooling. However, in order to do so, there are a number of questions and challenges to consider, and these will form the focus of the workshop.

Workshop Scope

Key questions, to be addressed through position papers and discussion:

- What, if any, are the precursors to computer science skills and understanding and how can we foster them?
- Can we draw on knowledge from other subject areas (such as mathematics) where conceptual pathways seem more clearly understood?
- How can we ensure that elementary concepts are refined and deepened over time (e.g. similar to Bruner's spiral curriculum [2]).
- What is the relationship between programming and computational thinking and are there any trade-offs in terms of which should be the primary educational focus?
- How can we design programming tools and curricula that are developmentally appropriate and foster motivation throughout childhood?

Workshop schedule

- Position papers and discussion
- Hands on overview of current tools (during breaks)
- Road map activity: participants work in groups to draw a road map of the developmental stages in computational thinking.
- Whole group plenary: discussion and identification of gaps in knowledge and research in the field.

Expected outcomes

The workshop will bring together expertise from technology-enhanced learning, primary education, programming education, and novice language design. Attendees will be invited to submit to a special issue in the Journal of Child Computer Interaction, and a collaborative paper will develop the ideas from the road map activity.

Organisers

Kate Howland is a Lecturer in Interaction Design at the University of Sussex, and researches the design of creative technologies for young people, particularly novice programming languages, and tools for interactive storytelling and game design.

Judith Good is a Reader in Informatics at the University of Sussex. Her research focuses on the development of motivating tools, languages and environments for novice programmers, particularly at the secondary school level, but increasingly at the primary school level as well.

Judy Robertson is Professor of Digital Learning at the Moray House School of Education. She is interested in Computer Science (CS) education and the benefits of creative game making in children. She is on the Royal Society of Edinburgh advisory group to develop new CS curriculum materials for schools, and is currently

evaluating the impact of an innovative approach to CS teacher development.

Andrew Manches was previously an infant teacher and is now a Chancellor's Fellow at the University of Edinburgh researching the role of interaction in early conceptual development. Andrew is also the director of a start-up that has designed, built and now sells an early learning computing education device.

References

[1] Bers, M.U., L. Flannery, E.R. Kazakoff, and A. Sullivan, *Computational thinking and tinkering: Exploration of an early childhood robotics curriculum.* Computers & Education, 2014. **72**: p. 145-157.

[2] Bruner, J.S., *The process of education*1960, Oxford, England: Harvard University Press, available as e-book (2009).

[3] Carey, S. and E. Spelke, *Domain-specific knowledge and conceptual change.* Mapping the mind: Domain specificity in cognition and culture, 1994: p. 169-200.

[4] Computer Science Teachers Association. *Computer Science K–8: Building a Strong Foundation*. 2012; Available from: http://csta.acm.org/Curriculum/sub/CurrFiles/CS_K-8_Building_a_Foundation.pdf.

[5] Dann, W.P., S. Cooper, and R. Pausch, *Learning to Program with Alice (w/CD ROM)*2011: Prentice Hall Press.

[6] Department for Education. *National curriculum in England: computing programmes of study*. 2013; Available from: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

[7] Flannery, L.P., B. Silverman, E.R. Kazakoff, M.U. Bers, P. Bontá, and M. Resnick. *Designing ScratchJr: support for early childhood learning through computer programming*. in *Proceedings of IDC 2013*. 2013. ACM.

[8] Frei, P., V. Su, B. Mikhak, and H. Ishii. *Curlybot: designing a new class of computational toys*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2000. ACM.

[9] Howland, K. and J. Good, *Learning to communicate computationally with Flip: A bi-modal programming language for game creation.* Computers & Education, 2015. **80**(0): p. 224-240.

[10] Jacobsen, H. *Five-years-olds learn coding in schools to prepare for future labour market*. EurActiv.com - EU News & policy debates, across languages, 2014.

[11] Kölling, M., *The greenfoot programming environment.* ACM Transactions on Computing Education (TOCE), 2010. **10**(4): p. 14.

[12] Piaget, J., *Intellectual Evolution from Adolescence to Adulthood*, in *Cognitive and moral development and academic achievement in adolescence (originally published in Human development 15.1 (1972): 1-12.)*, R.M. Lerner and J. Jovanovic, Editors. 1999, Taylor & Francis.

[13] Raffle, H.S., A.J. Parkes, and H. Ishii. *Topobo: a constructive assembly system with kinetic memory*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2004. ACM.

[14] Resnick, M., J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, and B. Silverman, *Scratch: programming for all.* Communications of the ACM, 2009. **52**(11): p. 60-67.

[15] Resnick, M., F. Martin, R. Berg, R. Borovoy, V. Colella, K. Kramer, and B. Silverman. *Digital manipulatives: new toys to think with*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1998.

[16] Vihavainen, A., J. Airaksinen, and C. Watson. *A systematic review of approaches for teaching introductory programming and their influence on success*. in *Proceedings of the tenth annual conference on International computing education research*. 2014. ACM.

[17] Wyeth, P. and H.C. Purchase, *Using developmental theories to inform the design of technology for children*, in *Proceedings of the 2003 conference on Interaction design and children*2003, ACM. p. 93-100.